

LEASE MANAGEMENT

COLLEGE NAME: Sri Ramakrishna College Of Arts And Science
For Women

COLLEGE CODE: bru28

TEAM ID: NM2025TMID24206

TEAM MEMBERS:

TEAM LEADER : Ezhilmozhi VJ

EMAIL ID : srcw2322k112@srcw.ac.in

TEAM MEMBER1 : Dharshini M

EMAIL ID : srcw2322k109@srcw.ac.in

TEAM MEMBER2 : Harini H

EMAIL ID : srcw2322k113@srcw.ac.in

TEAM MEMBER3 : Divya V

EMAIL ID : srcw2322k111@srcw.ac.in

Introduction

Lease Management is the process of handling agreements between a property owner (landlord) and a tenant. It involves maintaining property records, managing tenant details, tracking lease start and end dates, ensuring timely rent payments, and automating approvals. In Salesforce, lease management can be simplified using custom objects (Property, Tenant, Lease, Payment), workflows, approval processes, triggers, and email notifications.



Purpose

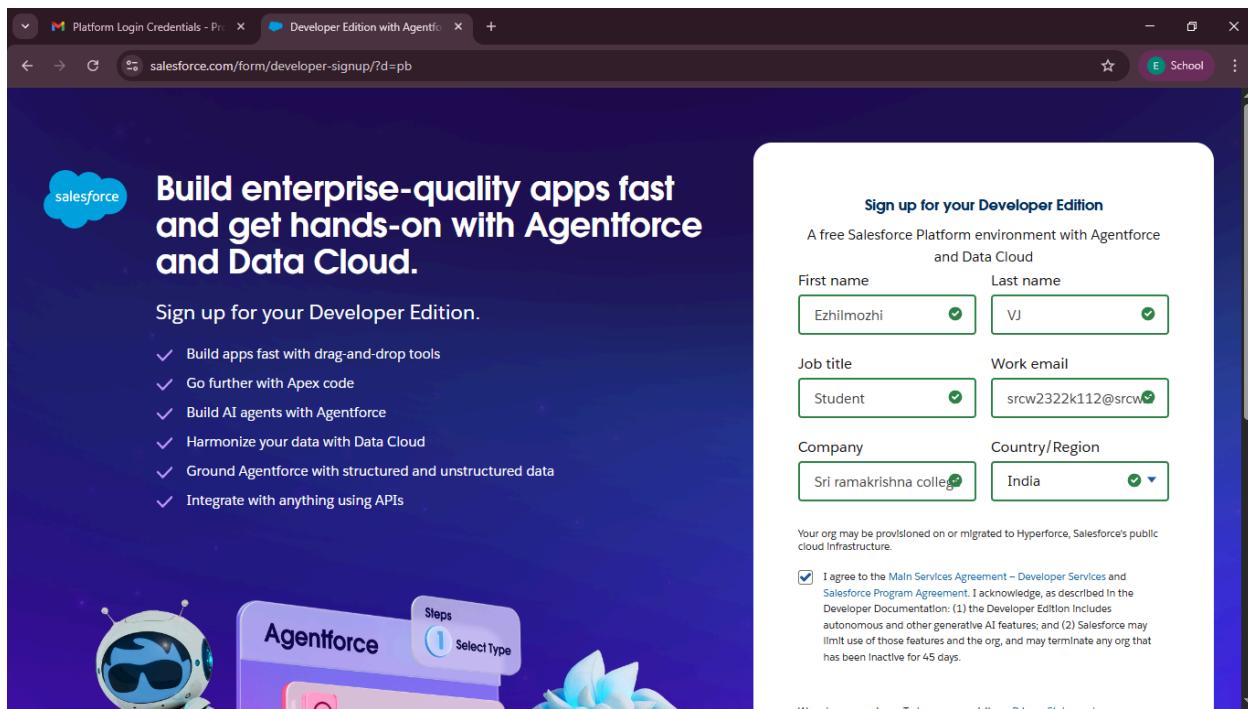
- To centralize property, tenant, and lease information.

- To automate approval of lease requests and tenant leaving requests.
- To ensure compliance by validating lease dates (e.g., End Date > Start Date).
- To send timely notifications for approvals, rejections, and payments.
- To track monthly rent payments with reminders and receipts.
- To improve transparency between landlords, tenants, and administrators.

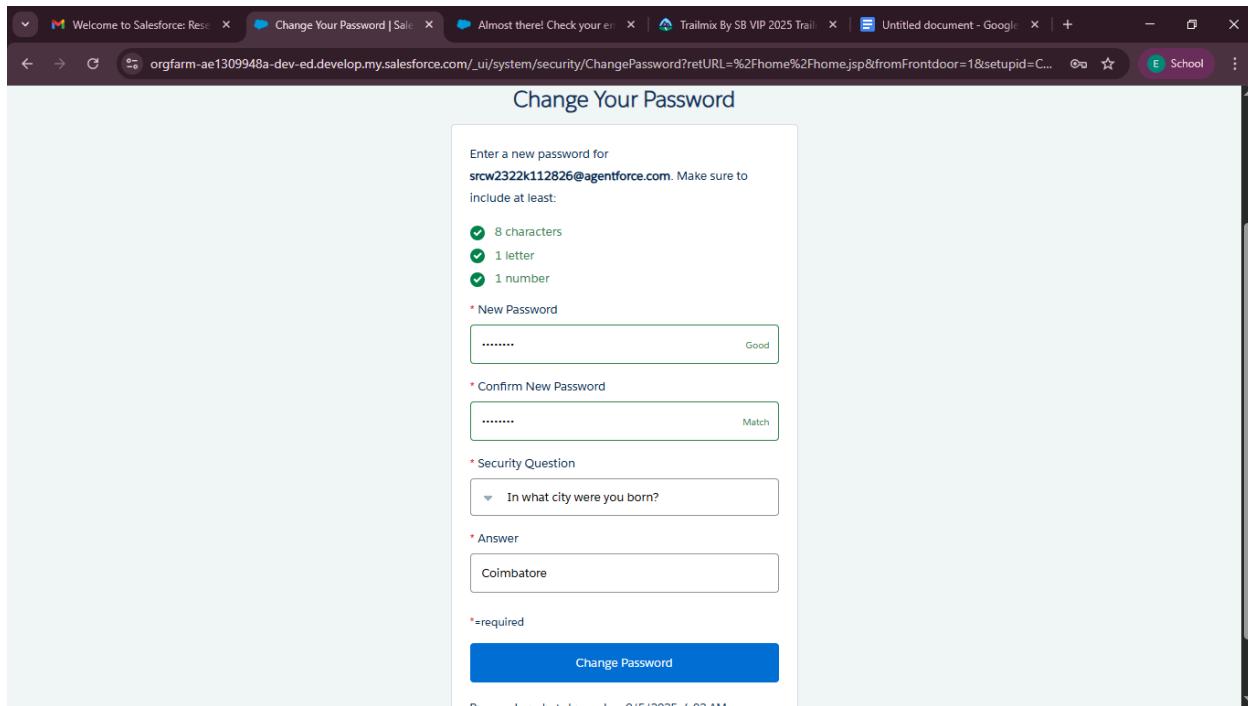
DEVELOPMENT PHASE:

The development phase of the Lease Management system was carried out in 11 structured milestones to ensure smooth progress and systematic completion. It began with creating the custom objects (Property, Tenant, Lease, and Payment) followed by setting up their tabs and a Lightning App to bring them together. Next, fields and lookup relationships were added to establish data connections, and validation rules like ensuring the end date is greater than the start date were implemented for accuracy. In the following milestones, email templates were designed for tenant leaving, approvals, rejections, and payments. Two approval processes were built to manage tenant requests and vacancy checks. The phase also included writing Apex triggers and handlers with test classes to automate critical actions, and flows for handling monthly payments. Finally, a scheduled class was developed for recurring tasks, completing the 11 milestones of development. Each step contributed to building a robust and automated lease management system.

Creating Developer Account:



Account Activation:



OBJECT CREATION:

The screenshot shows the Salesforce Setup interface with the 'Object Manager' tab selected. A new object named 'Property' is being created. The 'Details' section includes fields for API Name (Property__c), Singular Label (Property), and Plural Label (Properties). The 'Fields & Relationships' sidebar lists various configuration options like Page Layouts, Lightning Record Pages, and Buttons, Links, and Actions.

Details	
Description	
API Name	Property__c
Custom	✓
Singular Label	Property
Plural Label	Properties
Enable Reports	✓
Track Activities	✓
Track Field History	✓
Deployment Status	Deployed
Help Settings	Standard salesforce.com Help Window

The screenshot shows the Salesforce Setup interface with the 'Object Manager' tab selected. A new object named 'Tenant' is being created. The 'Details' section includes fields for API Name (Tenant__c), Singular Label (Tenant), and Plural Label (Tenants). The 'Fields & Relationships' sidebar lists various configuration options like Page Layouts, Lightning Record Pages, and Buttons, Links, and Actions.

Details	
Description	
API Name	Tenant__c
Custom	✓
Singular Label	Tenant
Plural Label	Tenants
Enable Reports	✓
Track Activities	✓
Track Field History	✓
Deployment Status	Deployed
Help Settings	Standard salesforce.com Help Window

The screenshot shows the Salesforce Setup interface for the 'Object Manager'. The top navigation bar includes tabs for 'Platform Login Credentials', '- Student', 'Home | Salesforce', 'lease | Salesforce', and 'Untitled document - Google'. The main title is 'SETUP > OBJECT MANAGER' followed by the object name 'lease'. The left sidebar lists various configuration options: Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, and List View Button Layout. The right panel displays the 'Details' section for the 'lease' object. It includes fields for Description, API Name (set to 'lease__c'), Singular Label (set to 'lease'), Plural Label (set to 'lease'), and several checkboxes for reporting and tracking features like 'Enable Reports', 'Track Activities', 'Track Field History', and 'Deployment Status' (set to 'Deployed'). There are also links for 'Help Settings' and 'Standard salesforce.com Help Window'. At the bottom right are 'Edit' and 'Delete' buttons.

The screenshot shows the Salesforce Setup interface for the 'Object Manager'. The top navigation bar includes tabs for 'Platform Login Credentials', '- Student', 'Home | Salesforce', 'lease | Salesforce', and 'Untitled document - Google'. The main title is 'SETUP > OBJECT MANAGER' followed by the object name 'Payment for tenanat'. The left sidebar lists various configuration options: Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, and List View Button Layout. The right panel displays the 'Details' section for the 'Payment for tenanat' object. It includes fields for Description, API Name (set to 'Payment_for_tenanat__c'), Singular Label (set to 'Payment for tenanat'), Plural Label (set to 'Payment for tenants'), and several checkboxes for reporting and tracking features like 'Enable Reports', 'Track Activities', 'Track Field History', and 'Deployment Status' (set to 'Deployed'). There are also links for 'Help Settings' and 'Standard salesforce.com Help Window'. At the bottom right are 'Edit' and 'Delete' buttons.

TABS:

The screenshot shows the Salesforce Setup interface with the 'Tabs' page selected. The left sidebar has 'User Interface' expanded, with 'Tabs' selected under 'Rename Tabs and Labels'. The main content area displays sections for 'Custom Object Tabs', 'Web Tabs', and 'Visualforce Tabs'. Under 'Custom Object Tabs', there are four entries: 'lease' (Tab Style: Books), 'Payment' (Tab Style: Books), 'Properties' (Tab Style: Books), and 'Tenants' (Tab Style: Books). A note at the top says: 'You can create new custom tabs to extend Salesforce functionality or to build new application functionality. Custom Object tabs look and behave like the standard tabs provided with Salesforce. Web tabs allow you to embed external web applications and content within the Salesforce window. Visualforce tabs allow you to embed Visualforce pages. Lightning Component tabs allow you to add Lightning components to the navigation menu in Lightning Experience and the mobile app. Lightning Page tabs allow you to add Lightning Pages to Lightning Experience and the mobile app.' A 'Help for this Page' link is also present.

FIELDS AND RELATIONSHIP:

The screenshot shows the Salesforce Setup interface with the 'Fields & Relationships' page for the 'Property' object selected. The left sidebar has 'Object Manager' selected under 'SETUP'. The main content area shows a table of fields and relationships. The table has columns: FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The data includes:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address__c	Long Text Area(32768)		
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Name	Name__c	Text(18)		
Owner	OwnerId	Lookup(User,Group)	✓	
Property Name	Name	Text(80)	✓	
sfqt	sfqt__c	Text(18)		
Type	Type__c	Picklist		

Upload files - Ezhilmozhi18/Lease | Documentation - Google Docs | Payment for tenantat | Salesforce

orgfarm-ae1309948a-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/01lgK0000022pQH/FieldsAndRelationships/view

Cloud icon, Search Setup, Home, Object Manager, Fields & Relationships

SETUP > OBJECT MANAGER
Payment for tenantat

Details, Fields & Relationships (9 items)

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Amount	Amount__c	Number(18, 0)		
check for payment	check_for_payment__c	Picklist		
Created By	CreatedBy	Lookup(User)		
Last Modified By	LastModifiedBy	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Payment date	Payment_date__c	Date		
Payment Name	Name	Text(80)		✓
Property	Property__c	Lookup(Property)		✓

Upload files - Ezhilmozhi18/Lease | Documentation - Google Docs | lease | Salesforce

orgfarm-ae1309948a-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/01lgK0000022pV7/FieldsAndRelationships/view

Cloud icon, Search Setup, Home, Object Manager, Fields & Relationships

SETUP > OBJECT MANAGER
lease

Details, Fields & Relationships (7 items)

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		
End date	End_date__c	Date		
Last Modified By	LastModifiedBy	Lookup(User)		
lease Name	Name	Text(80)		✓
Owner	OwnerId	Lookup(User,Group)		✓
Property	Property__c	Lookup(Property)		✓
start date	start_date__c	Date		

The screenshot shows the Salesforce Object Manager interface. The left sidebar has a 'Fields & Relationships' section selected. The main area displays a table titled 'Fields & Relationships' with 8 items, sorted by Field Label. The columns are FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED.

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Email	Email__c	Email		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)	✓	
Phone	Phone__c	Phone		
Property	Property__c	Lookup(Property)	✓	
status	status__c	Picklist		
Tenant Name	Name	Text(80)	✓	

THE LIGHTNING APP:

The screenshot shows the Lightning App Builder interface. The left sidebar has an 'App Details & Branding' section selected. The main area displays 'App Details & Branding' settings. It includes fields for App Name (Lease Management), Developer Name (Lease_Management), Description (Enter a description...), Image (Upload button), Primary Color Hex Value (#0070D2), and Org Theme Options (checkbox for using the app's image and color instead of the org's custom theme). Below this is an 'App Launcher Preview' section showing a preview of the app icon and name.

Lease Management - Lightning

Lightning App Builder App Settings Pages Lease Management

App Settings

App Details & Branding
App Options
Utility Items (Desktop Only)

Navigation Items

User Profiles

Navigation Items

Choose the items to include in the app, and arrange the order in which they appear. Users can personalize the navigation to add or move items, but users can't remove or rename the items that you add. Some navigation items are available only for phone or only for desktop. These items are dropped from the navigation bar when the app is viewed in a format that the item doesn't support.

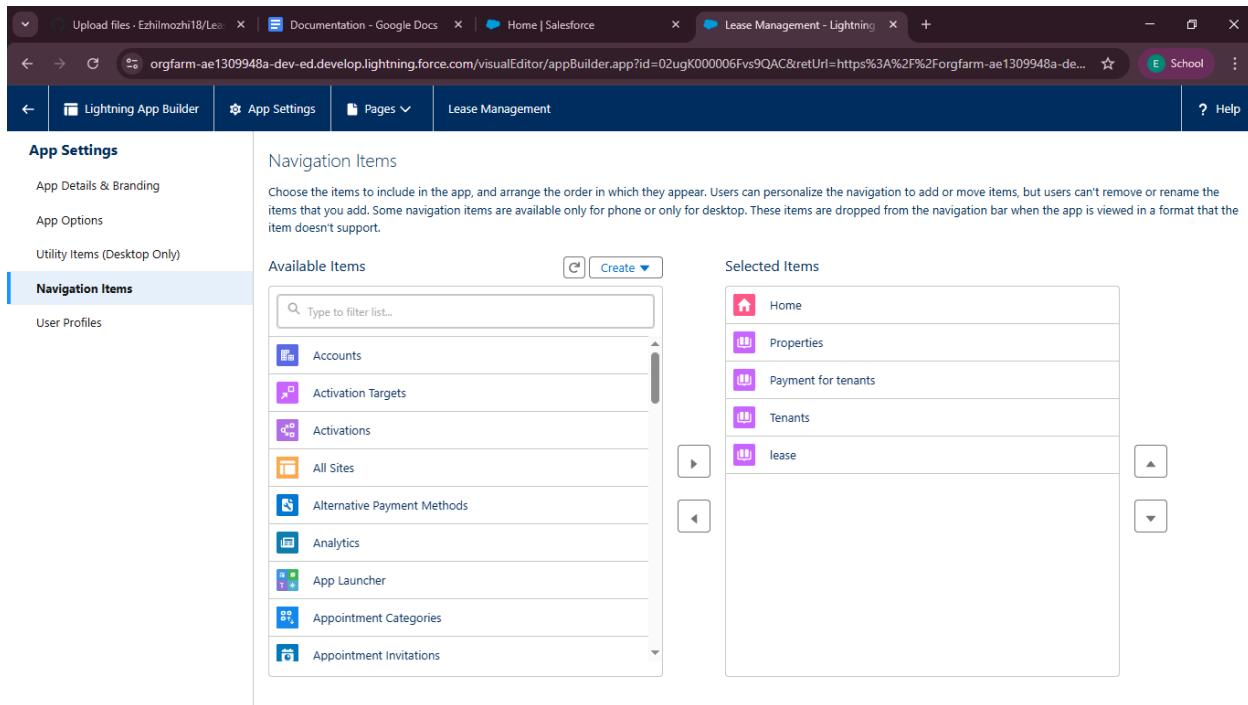
Available Items

Type to filter list...

Accounts
Activation Targets
Activations
All Sites
Alternative Payment Methods
Analytics
App Launcher
Appointment Categories
Appointment Invitations

Selected Items

Home
Properties
Payment for tenants
Tenants
Lease



Lease Management - Lightning

Lightning App Builder App Settings Pages Lease Management

App Settings

App Details & Branding
App Options
Utility Items (Desktop Only)

User Profiles

Navigation Items

User Profiles

Choose the user profiles that can access this app.

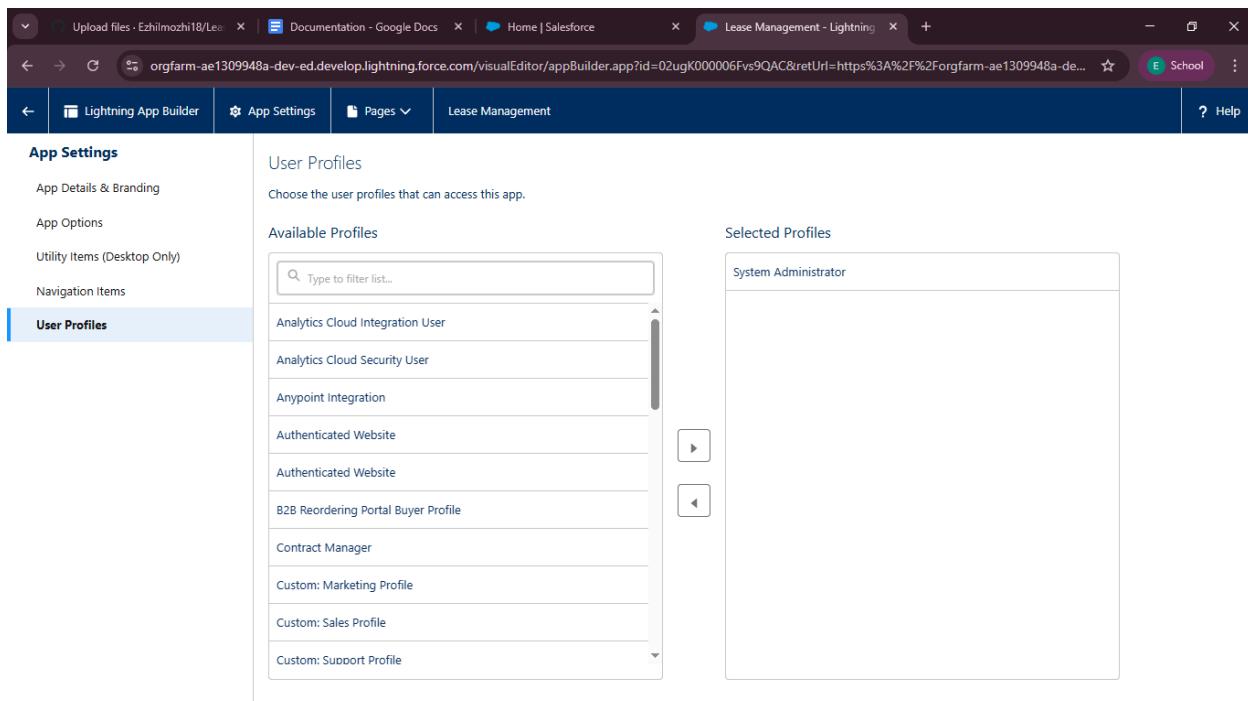
Available Profiles

Type to filter list...

Analytics Cloud Integration User
Analytics Cloud Security User
Anypoint Integration
Authenticated Website
Authenticated Website
B2B Reordering Portal Buyer Profile
Contract Manager
Custom: Marketing Profile
Custom: Sales Profile
Custom: Support Profile

Selected Profiles

System Administrator



VALIDATION RULE:

The screenshot shows the Salesforce Setup interface for the 'lease' object. The left sidebar lists various setup options like Details, Fields & Relationships, Page Layouts, and Lightning Record Pages. The main content area displays the 'Validation Rule Detail' for the 'lease_end_date' rule. The rule details are as follows:

Field	Value
Rule Name	lease_end_date
Error Condition Formula	End_date__c > start_date__c
Error Message	Your End date must be greater than start date
Description	
Created By	Ezhilmozhi V.J. 9/5/2025, 6:40 AM
Modified By	Ezhilmozhi V.J. 9/5/2025, 6:40 AM

EMAIL TEMPLATES:

Classic Email Templates

Tenant Email

Preview your email template below.

Email Template Detail

Email Templates from Salesforce	Unfiled Public Classic Email Templates
Email Template Name	Tenant Email
Template Unique Name	Tenant_Email
Encoding	Unicode (UTF-8)
Author	Ezhilmozhi VJ [Change]
Description	
Created By	Ezhilmozhi VJ, 9/5/2025, 6:52 AM
Modified By	Ezhilmozhi VJ, 9/6/2025, 4:21 AM

Email Template

Send Test and Verify Merge Fields

Subject | Urgent: Monthly Rent Payment Reminder

Plain Text Preview

Dear {Tenant__c.Name},
I trust this email finds you well. We appreciate your continued tenancy at our property and I hope you have been comfortable

Classic Email Templates

tenant payment

Preview your email template below.

Email Template Detail

Email Templates from Salesforce	Unfiled Public Classic Email Templates
Email Template Name	tenant payment
Template Unique Name	tenant_payment
Encoding	Unicode (UTF-8)
Author	Ezhilmozhi VJ [Change]
Description	
Created By	Ezhilmozhi VJ, 9/5/2025, 6:54 AM
Modified By	Ezhilmozhi VJ, 9/6/2025, 4:21 AM

Email Template

Send Test and Verify Merge Fields

Subject | Confirmation of Successful Monthly Payment

Plain Text Preview

Dear {Tenant__c.Email__c},
We hope this email finds you well. We are writing to inform you that we have successfully received your monthly payment.

Screenshot of the Salesforce Setup interface showing the "Classic Email Templates" page for the "Leave rejected" template.

Email Template Detail

Email Templates from Salesforce	Unfiled Public Classic Email Templates
Email Template Name	Leave rejected
Template Unique Name	Leave_rejected
Encoding	Unicode (UTF-8)
Author	Ezhilmozhi VJ [Change]
Description	
Created By	Ezhilmozhi VJ 9/5/2025, 6:50 AM
Modified By	Ezhilmozhi VJ 9/5/2025, 7:31 AM

Email Template

Subject: Leave rejected

Plain Text Preview:

```
Dear {!Tenant__c.Name},  
  
I hope this email finds you well. Your contract has not ended. So we can't approve your leave.
```

Screenshot of the Salesforce Setup interface showing the "Classic Email Templates" page for the "Leave approved" template.

Email Template Detail

Email Templates from Salesforce	Unfiled Public Classic Email Templates
Email Template Name	Leave approved
Template Unique Name	Leave_approved
Encoding	Unicode (UTF-8)
Author	Ezhilmozhi VJ [Change]
Description	
Created By	Ezhilmozhi VJ 9/5/2025, 6:47 AM
Modified By	Ezhilmozhi VJ 9/5/2025, 7:28 AM

Email Template

Subject: Leave approved

Plain Text Preview:

```
dear {!Tenant__c.Name},  
  
I hope this message finds you well. I am writing to inform you that I have received your email confirming the approval of my
```

Classic Email Templates

tenant leaving

Preview your email template below.

Email Template Detail

Email Template Name	Unfiled Public Classic Email Templates
Template Unique Name	tenant_leaving
Encoding	Unicode (UTF-8)
Author	Ezhilmozhi VJ [Change]
Description	
Created By	Ezhilmozhi VJ 9/5/2025, 6:44 AM
Modified By	Ezhilmozhi VJ 9/5/2025, 7:22 AM

Email Template

Subject: request for approve the leave

Plain Text Preview:

Dear {Tenant__c.CreatedBy},
Please approve my leave

APPROVAL PROCESS:

Approval Processes

Tenant: check for vacant

Process Definition Detail

Process Name	check for vacant	Active
Unique Name	check_for_vacant	Next Automated Approver Determined By
Description		
Entry Criteria	Tenant: status NOTEQUALTO Leaving	
Record Editability	Administrator ONLY	Allow Submitters to Recall Approval Requests
Approval Assignment Email Template	Tenant Owner	
Initial Submitters		
Created By	Ezhilmozhi VJ 9/5/2025, 7:14 AM	Modified By Ezhilmozhi VJ 9/6/2025, 5:10 AM

Initial Submission Actions

Action	Type	Description
Record Lock		Lock the record from being edited
Email Alert		Please approve my leave

Approval Steps

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
Show Actions Edit	1	Submit for approval		Tenant: status EQUALS StayLeaving , else Approve	Manually Chosen	Final Rejection

Final Approval Actions

Action	Type	Description
Edit	Record Lock	Lock the record from being edited
Edit Remove	Email Alert	Tenant leaving

Final Rejection Actions

Action	Type	Description
Edit	Record Lock	Unlock the record for editing
Edit Remove	Email Alert	your request for leave is rejected

Recall Actions

Action	Type	Description
	Record Lock	Unlock the record for editing

APEX TRIGGER:

```

trigger test on Tenant__c (before insert)
{
    if(trigger.isInsert && trigger.isBefore){
        testHandler.preventInsert(trigger.new);
    }
}

```

Class	Percent	Lines
Overall	0%	
MonthlyEmailScheduler	0%	0/15
test	0%	0/2
testHandler	0%	0/7

Developer Console - Google Chrome
 orgfarm-ae1309948a-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

```

File Edit Debug Test Workspace Help < >
test.aprx testHandler.apxc
Code Coverage: None API Version: 64 Go To
1 public class testHandler {
2
3   public static void preventInsert(List<Tenant__c> newList) {
4
5     Set<Id> existingPropertyIds = new Set<Id>();
6
7     for (Tenant__c existingTenant : [SELECT Id, Property__c FROM Tenant__c WHERE Property__c != null]) {
8
9       existingPropertyIds.add(existingTenant.Property__c);
10
11     }
12
13
14   for (Tenant__c newTenant : newList) {
15
16
17
  
```

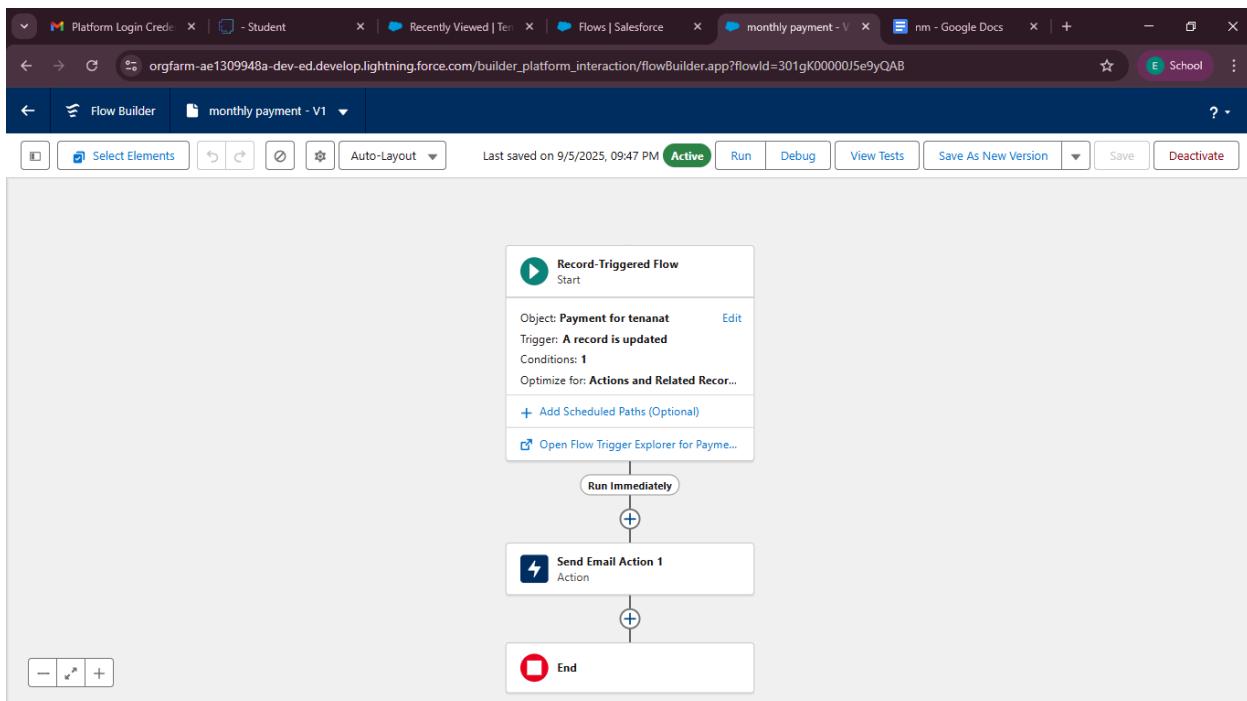
Logs Tests Checkpoints Query Editor View State Progress Problems

Status	Test Run	Enqueued Time	Duration	Failures	Total

Overall Code Coverage

Class	Percent	Lines
Overall	0%	
MonthlyEmailScheduler	0%	0/15
test	0%	0/2
testHandler	0%	0/7

FLows:



Flow Builder | monthly payment - V1

Last saved on 9/5/2025, 09:47 PM Active Run Debug View Tests Save As New Version Save Deactivate

Record-Triggered Flow Start

Object: Payment for tenant Edit
Trigger: A record is updated
Conditions: 1
Optimize for: Actions and Related Recor...
+ Add Scheduled Paths (Optional)
Open Flow Trigger Explorer for Payne...

Run Immediately

Send Email Action 1 Action

End

Configure Start

Select Object
Select the object whose records trigger the flow when they're created, updated, or deleted.
* Object
Payment for tenant

Configure Trigger
Trigger the Flow When:
 A record is created
 A record is updated
 A record is created or updated
 A record is deleted

Set Entry Conditions
Specify entry conditions to reduce the number of records that trigger the flow and the number of times the flow is executed. Minimizing unnecessary flow executions helps to conserve your org's resources.
If you create a flow that's triggered when a record is updated, we recommend first defining entry conditions. Then select the **Only when a record is updated** to meet the **condition requirements** option for When to Run the Flow for Updated Records.

Condition Requirements
All Conditions Are Met (AND)

```

graph TD
    Start((Record-Triggered Flow Start)) --> Run[Run Immediately]
    Run --> Action[Send Email Action 1]
    Action --> End((End))
  
```

Flow Builder | monthly payment - V1

Last saved on 9/5/2025, 09:47 PM Active Run Debug View Tests Save As New Version Save Deactivate

Record-Triggered Flow Start

Object: Payment for tenant Edit
Trigger: A record is updated
Conditions: 1
Optimize for: Actions and Related Recor...
+ Add Scheduled Paths (Optional)
Open Flow Trigger Explorer for Payne...

Run Immediately

Send Email Action 1 Action

End

Configure Start

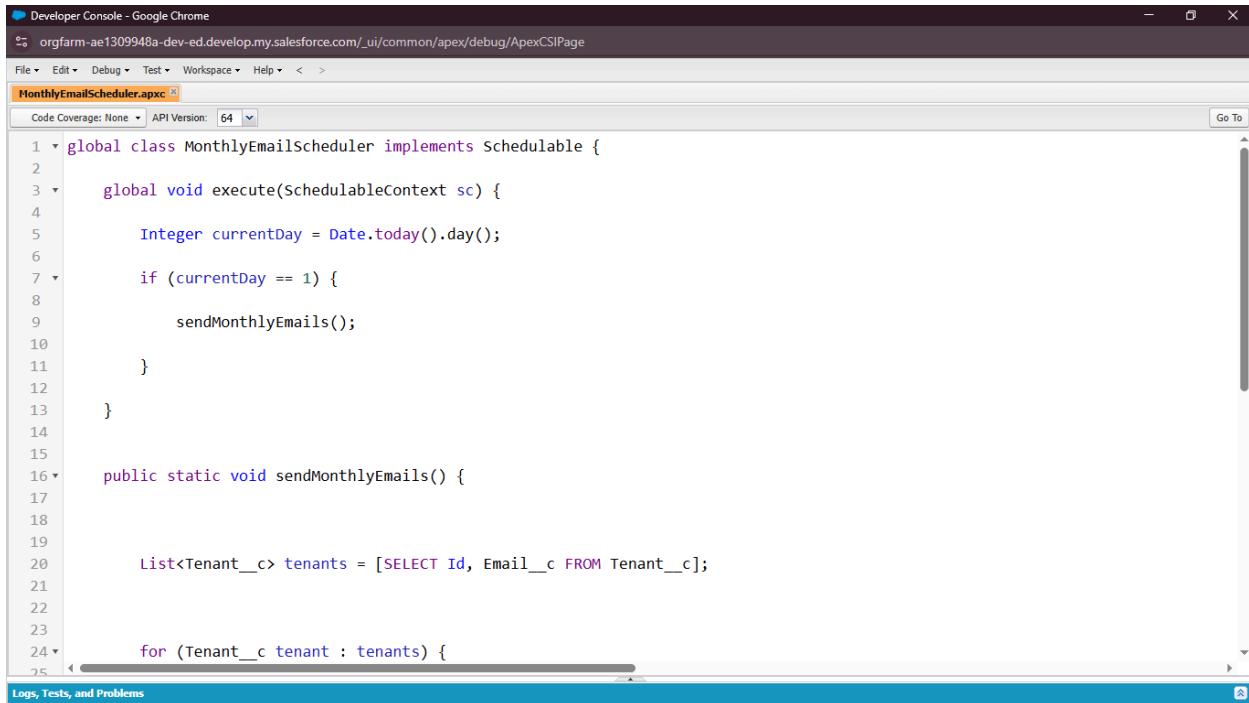
Field: check for payment Operator: Equals Value: Paid
+ Add Condition

When to Run the Flow for Updated Records
 Every time a record is updated and meets the condition requirements
 Only when a record is updated to meet the condition requirements

Optimize Flow
Optimize the Flow for:
Fast Field Updates: Update fields on the record that triggers the flow to run. This high-performance flow runs **before the record is saved** to the database.
Actions and Related Records: Update any record and perform actions, like send an email. This more flexible flow runs **after the record is saved** to the database.

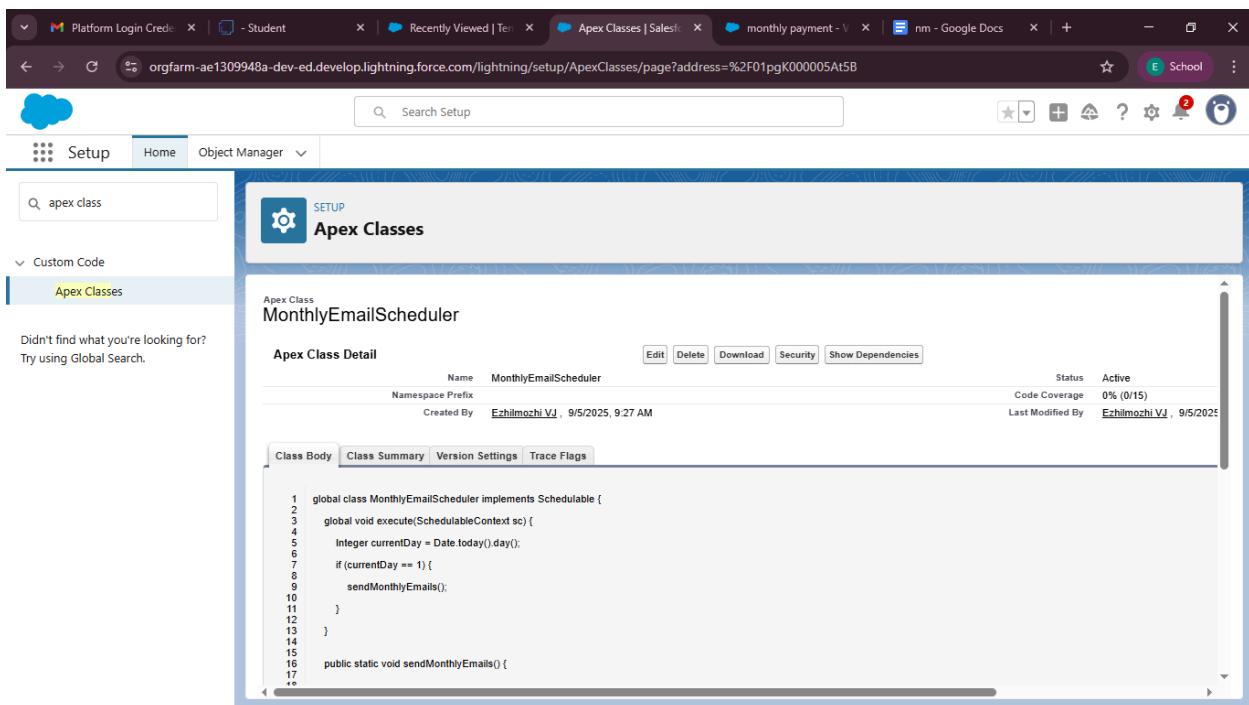
Is this flow making an external callout or connecting to an external system?
An asynchronous path is required for flows that involve external systems.

SCHEDULE CLASSES:



The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is orgfarm-ae1309948a-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage. The page displays the code for the Apex class `MonthlyEmailScheduler`. The code implements the `Schedulable` interface and contains logic to send monthly emails if the current day is 1.

```
1 global class MonthlyEmailScheduler implements Schedulable {  
2     global void execute(SchedulableContext sc) {  
3         Integer currentDay = Date.today().day();  
4         if (currentDay == 1) {  
5             sendMonthlyEmails();  
6         }  
7     }  
8     public static void sendMonthlyEmails() {  
9         List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];  
10        for (Tenant__c tenant : tenants) {  
11            // Email sending logic  
12        }  
13    }  
14}  
15  
16 public static void sendMonthlyEmails() {  
17 }
```



The screenshot shows the Salesforce Setup Apex Classes page. The URL is orgfarm-ae1309948a-dev-ed.develop.lightning.force.com/lightning/setup/ApexClasses/page?address=%2F01pgK000005At5B. The page displays the details for the Apex Class `MonthlyEmailScheduler`. The class is active and was created by Ezhilmozhi VJ on 9/5/2025, 9:27 AM. The code body is identical to the one shown in the Developer Console.

Apex Class Detail

Name	Namespace Prefix	Status	Active
MonthlyEmailScheduler		Code Coverage 0% (0/15)	0% (0/15)
		Last Modified By	Ezhilmozhi VJ , 9/5/2025

Class Body

```
1 global class MonthlyEmailScheduler implements Schedulable {  
2     global void execute(SchedulableContext sc) {  
3         Integer currentDay = Date.today().day();  
4         if (currentDay == 1) {  
5             sendMonthlyEmails();  
6         }  
7     }  
8     public static void sendMonthlyEmails() {  
9         List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];  
10        for (Tenant__c tenant : tenants) {  
11            // Email sending logic  
12        }  
13    }  
14}
```

This screenshot shows a Salesforce Lightning record page for a tenant named "Ezhil". The page has a header with tabs: "Lease Management", "Home", "Properties", "Payment for tenants", "Tenants", and "lease". The main content area displays the tenant's details: Tenant Name (Ezhil), Email (ezhilmozhiveeramoorthy@gmail.com), Phone (empty), status (Stay), and Property (Hyd). The "Owner" field shows "Ezhilmozhi VJ". Below the details, it says "Last Modified By" and shows the same user information. On the right side, there is a sidebar titled "Activity" with buttons for "New Case", "New Lead", "Delete", "Clone", "Change Owner", "Printable View", "Sharing", and "Sharing Hierarchy". A dropdown menu is open, showing options like "Submit for Approval" and "Edit Lead".

This screenshot shows the same Salesforce Lightning record page for the tenant "Ezhil" after the submission process. A green success message at the top left states "Tenant was submitted for approval." The rest of the page content is identical to the first screenshot, showing the tenant's details and the activity sidebar.

The screenshot shows a Salesforce Lightning interface for 'Lease Management'. A process instance step titled 'Tenant Approval' is displayed, showing it is 'Approved'. The details pane shows the submitter (Ezhilmozhi VJ), date submitted (Sep 7, 2025), actual approver (Ezhilmozhi VJ), and assigned to (Ezhilmozhi VJ). The notifications sidebar lists several recent activity items related to tenant approvals.

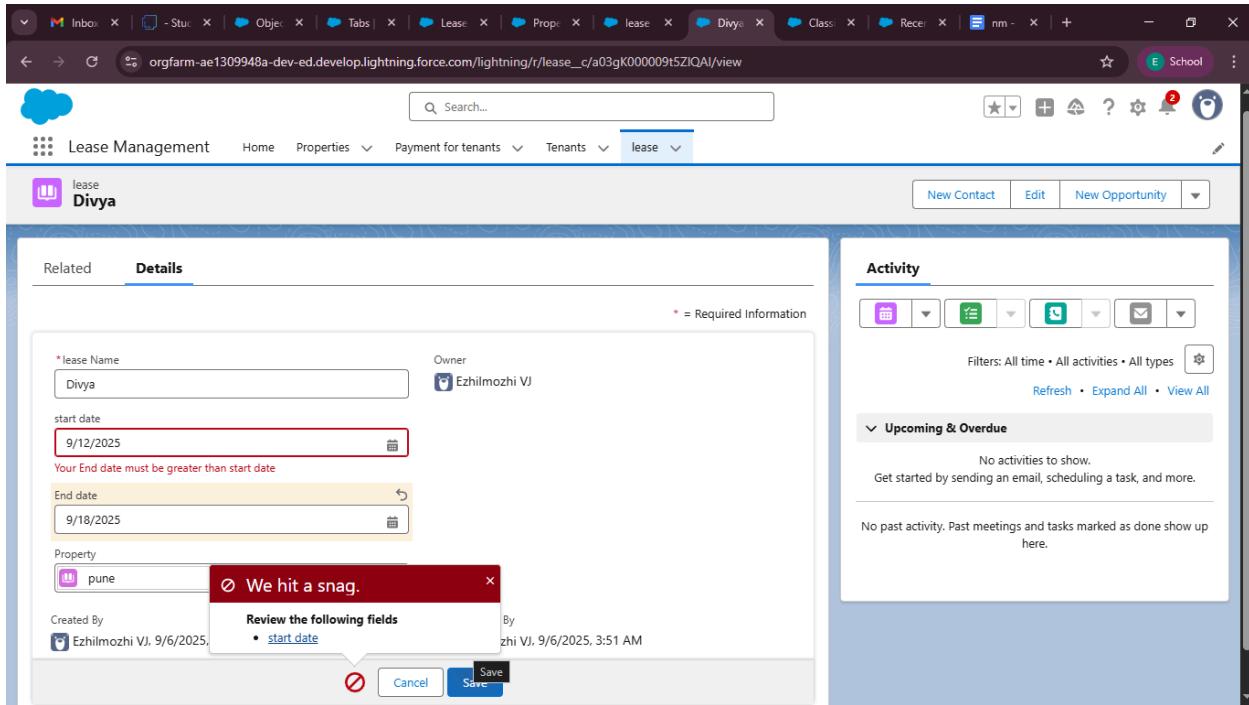
Notification	Details
Approval request for the tenant is approved Ezhil	a few seconds ago
Ezhilmozhi VJ is requesting approval for tenant Tenant Name: dd • Owner: Ezhilmozhi VJ	an hour ago
Approval request for the tenant is approved harini	2 hours ago
Approval request for the tenant is approved Ezhil	2 hours ago
Approval request for the tenant is rejected Chen	2 hours ago

FUNCTIONAL AND PERFORMANCE TESTING:

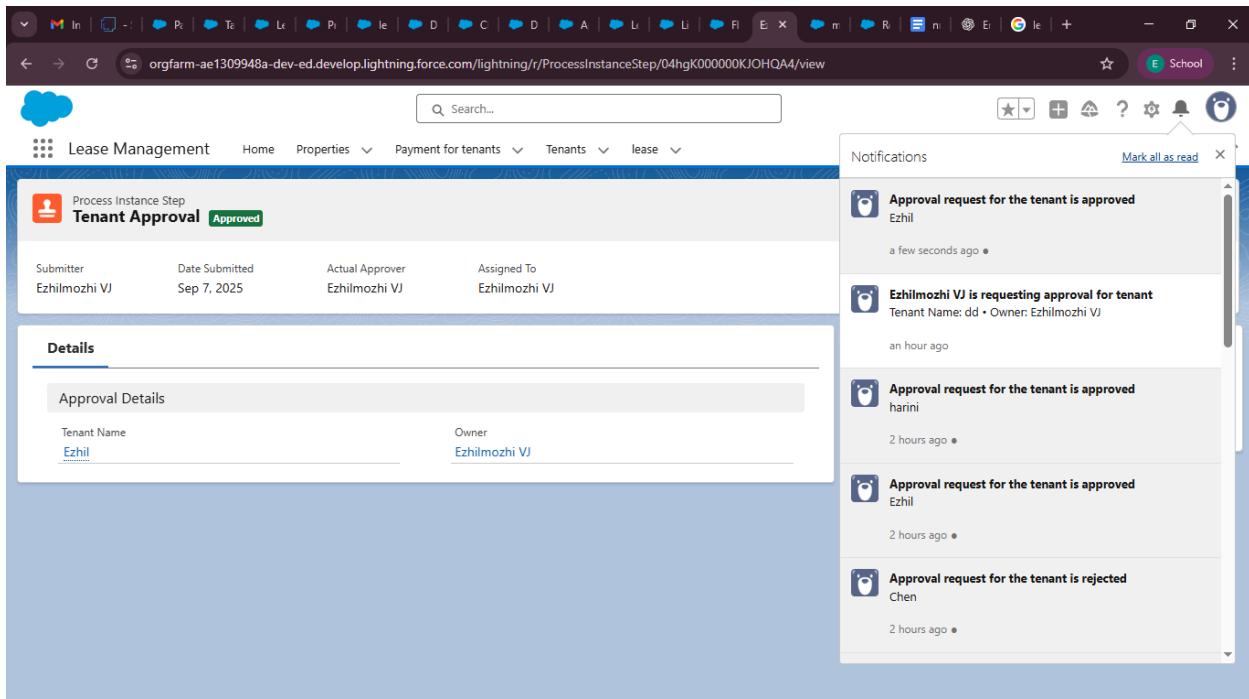
Trigger validation by entering duplicate tenant property records

The screenshot shows a 'New Tenant' form in the 'Lease Management' application. The 'Information' section includes fields for Tenant Name (Ezhil), Email (name@gmail.com), Phone (1234567890), status (Stay), and Property (Hyd). A validation error message 'We hit a snag.' is displayed, stating 'A tenant can have only one property'. The 'Save' button is highlighted in blue.

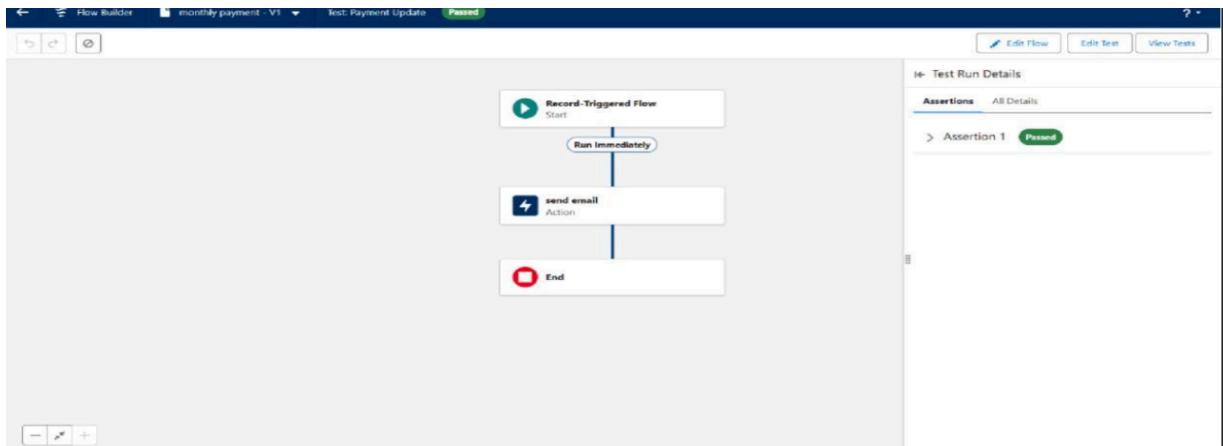
Validation rule checking:



Approval process validated through email alerts and status updates:



Test flows on payment update:



RESULTS

Output Screenshots

Tabs for Property, Tenants, Lease, Payment

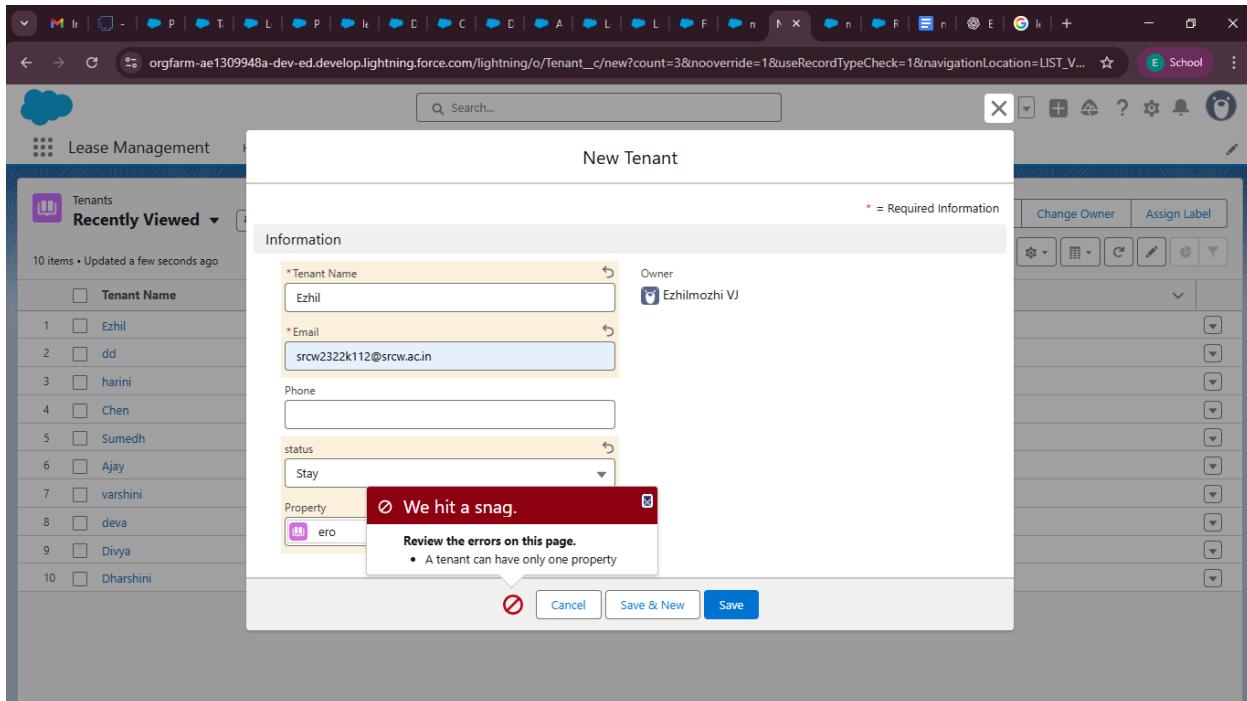
The screenshot shows the Salesforce Setup interface under the 'Tabs' section. The 'Custom Tabs' section is displayed, showing four custom object tabs: 'Lease' (Action: Edit | Del), 'Payment for tenants' (Action: Edit | Del), 'Properties' (Action: Edit | Del), and 'Tenants' (Action: Edit | Del). Each tab is styled with a purple 'Books' icon. The 'Tab Style' column indicates they are all set to 'Books'. The 'Description' column is empty. Below the custom tabs, there are sections for 'Web Tabs' and 'Visualforce Tabs', both of which currently have no entries.

Email alerts

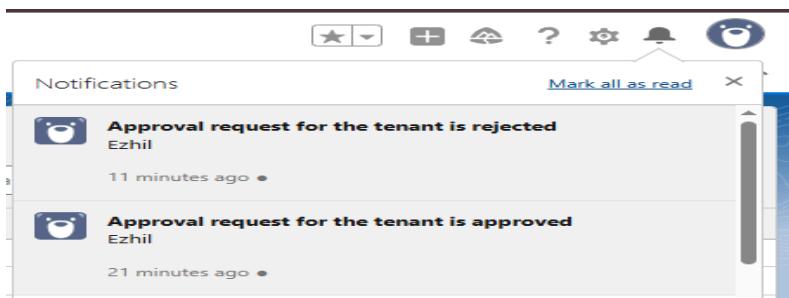
The screenshot shows the Salesforce Lease Management page for a tenant named 'Ezhilmozhi VJ'. The 'Approval History' section is displayed, showing a list of four approval steps:

Step Name	Date	Status	Assigned To	Actual Approver	Comments
1 Submit for approval	9/7/2025, 4:40 AM	Rejected	Ezhilmozhi VJ	Ezhilmozhi VJ	Rejected
2 Approval Request Submitted	9/7/2025, 4:40 AM	Submitted	Ezhilmozhi VJ	Ezhilmozhi VJ	Leaving
3 Submit for approval	9/7/2025, 4:31 AM	Approved	Ezhilmozhi VJ	Ezhilmozhi VJ	Approved
4 Approval Request Submitted	9/7/2025, 4:30 AM	Submitted	Ezhilmozhi VJ	Ezhilmozhi VJ	Leaving

Trigger error messages



Approval process notifications



CONCLUSION

Lease Management ensures smooth coordination between property owners, tenants, and administrators by centralizing records, automating approvals, and tracking payments. With the use of structured processes, validation rules, and timely notifications, it enhances transparency, reduces errors, and improves efficiency in managing property leases.

APPENDIX:

Test.apxt:

```
trigger test on Tenant__c (before insert)
{
    if(trigger.isInsert && trigger.isBefore){
        testHandler.preventInsert(trigger.new);
    }
}
```

testHandler.apxc:

```
public class testHandler {
    public static void preventInsert(List<Tenant__c> newlist) {
        Set<Id> existingPropertyIds = new Set<Id>();
        for (Tenant__c existingTenant : [SELECT Id, Property__c FROM Tenant__c
WHERE Property__c != null]) {
            existingPropertyIds.add(existingTenant.Property__c);
        }

        for (Tenant__c newTenant : newlist) {
            if (newTenant.Property__c != null &&
existingPropertyIds.contains(newTenant.Property__c)) {
                newTenantaddError('A tenant can have only one property');
            }
        }
    }
}
```

MonthlyEmailScheduler.apxc:

```
global class MonthlyEmailScheduler implements Schedulable {  
    global void execute(SchedulableContext sc) {  
        Integer currentDay = Date.today().day();  
        if (currentDay == 1) {  
            sendMonthlyEmails();  
        }  
    }  
  
    public static void sendMonthlyEmails() {  
  
        List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];  
  
        for (Tenant__c tenant : tenants) {  
            String recipientEmail = tenant.Email__c;  
            String emailContent = 'I trust this email finds you well. I am writing to remind  
you that the monthly rent is due. Your timely payment ensures the smooth  
functioning of our rental arrangement and helps maintain a positive living  
environment for all.';  
            String emailSubject = 'Reminder: Monthly Rent Payment Due';  
  
            Messaging.SingleEmailMessage email = new  
            Messaging.SingleEmailMessage();  
            email.setToAddresses(new String[]{recipientEmail});  
            email.setSubject(emailSubject);  
            email.setPlainTextBody(emailContent);  
  
            Messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});  
        }  
    }  
}
```